Coverage Extra Credit

iae225 — Extra Credit

Prompt

Code coverage relevant to a PR.

- Compute code coverage before the PR (on the merge base)
- Compute code coverage after the PR (on the revision PR currently is on)

For any modified line:

- say what the coverage was before and after the PR.
- say if coverage overall has improved or worsened.

WDYT?

Answer

I am going to analyze branch hw-4-v2.

This branch is directly off of the final pr for hw3

We added two types of integration tests in CircleCI. Unit tests are on the "*unit test*" job and integration and e2e both belong to the "*integration*" job.

Our CircleCI coverage is purely based on *unit tests*. It's odd because, in a way, it makes sense that all of the code of a component should be "covered" with the tests that that same component defines. This philosophy makes sense since the assertion "every component is responsible for testing all parts of itself" intuitively (at least to me) seems correct.

Preliminary Info

For our merge base we have (SHA 4277bfd2c872eb85f84c68ff953cbd7311c4671f), we can see that coverage is somewhat low:

```
Name                                                                                Stmts   Miss  Cover
-------------------------------------------------------------------------------------------------------
src/inbox_client_impl/src/inbox_client_impl/__init__.py                                 5      0   100%
src/inbox_client_impl/src/inbox_client_impl/_impl.py                                   63     20    68%
```

```
src/inbox_client_protocol/src/inbox_client_protocol/__init__.py       3      0   100%
src/message/src/message/__init__.py                                   1      0   100%
src/message_impl/src/message_impl/__init__.py                         5      0   100%
src/message_impl/src/message_impl/_impl.py                           80     24    70%
-----------------------------------------------------------------------------------
TOTAL                                                               157     44    72%
```

The files of interest are:
`src/inbox_client_impl/src/inbox_client_impl/_impl.py` with only 68% coverage
and
`src/message_impl/src/message_impl/_impl.py` with 70% coverage

Upon further analysis of these files, a mistake becomes clear in how we configured our coverage analysis.

When checking out head locally we get the following when running `pytest . --cov=src --cov-report=term-missing`:

```
src/inbox_client_impl/src/inbox_client_impl/__init__.py               5      0   100%
src/inbox_client_impl/src/inbox_client_impl/_impl.py                 63      9    86%   60-6
src/inbox_client_protocol/src/inbox_client_protocol/__init__.py       3      0   100%
src/message/src/message/__init__.py                                   1      0   100%
src/message_impl/src/message_impl/__init__.py                         5      0   100%
src/message_impl/src/message_impl/_impl.py                           80     24    70%   31-3
```

If we instead run `pytest . -m "not integration" --cov=src --cov-report=term-missing`:

```
Name                                                                Stmts   Miss  Cover   Miss
-----------------------------------------------------------------------------------------------
src/inbox_client_impl/src/inbox_client_impl/__init__.py               5      0   100%
src/inbox_client_impl/src/inbox_client_impl/_impl.py                 63     25    60%   60-6
src/inbox_client_protocol/src/inbox_client_protocol/__init__.py       3      0   100%
src/message/src/message/__init__.py                                   1      0   100%
src/message_impl/src/message_impl/__init__.py                         5      0   100%
src/message_impl/src/message_impl/_impl.py                           80     24    70%   31-3
-----------------------------------------------------------------------------------------------
TOTAL                                                               157     49    69%
```

You can see that the first command ends up with a much higher coverage of the inbox_client_impl component, specifically:

- CircleCI: 68%
- No Integration: 60%
- All Tests: 86%

  This error indicates a mistake with how we organized our CircleCI workflow.

Opinion

As mentioned previously, we split tests by unit and integration; however, we only ran coverage on unit tests. More specifically, some tests from integration (and potentially e2e) are raising our coverage locally, but then multiple tests do not get executed in CircleCI leading to lower coverage levels.

This indicates a fork in the road for "what philosophy do we want to take":

- A) We can mend together the two tests as a single "test" job on CircleCI, and have coverage run after all tests (unit, integration, and e2e) run.
- B) We can accept that we must test every part of the component within unit tests.

I believe that approach B is a better idea. *Components should be responsible for their own coverage.* It should not be the case that some parts of the component are only tested when integrating with another component.

> The idea of relying on a higher level of abstraction to pass our coverage tests seems and, likely is, incorrect.

Addendum: There is a further discrepancy caused by how we handle authentication in interactive (auth0 logins) vs environment (environment variables injected via CircleCI context) settings. This leads to tests running differently and we will explore this in Coverage Extra Credit > Analysis of Coverage Changes

Coverage changes upstream

After pulling in the component from the external team, CircleCI (expectedly) does not pass. Hence, the relevant changes come after our fixes dedicated specifically to coverage in the final PR for HW4 (SHA e8e4dcb7ab5c70031f572c8bc18707980c0caa8d)

Here are the 3 relevant coverage reports:

CircleCI:

```
src/inbox_client_impl/src/inbox_client_impl/__init__.py              10      2    80%
src/inbox_client_impl/src/inbox_client_impl/_impl.py                 88     14    84%
src/inbox_client_protocol/src/inbox_client_protocol/__init__.py       3      0   100%
src/message/src/message/__init__.py                                   1      0   100%
src/message_impl/src/message_impl/__init__.py                         5      0   100%
src/message_impl/src/message_impl/_impl.py                           80     24    70%
-------------------------------------------------------------------------------------
TOTAL                                                               187     40    79%
```

All Tests (- minus) Integration tests (local):

```
src/inbox_client_impl/src/inbox_client_impl/__init__.py              10      3    70%   18-2
src/inbox_client_impl/src/inbox_client_impl/_impl.py                 88     26    70%   46-4
src/inbox_client_protocol/src/inbox_client_protocol/__init__.py       3      0   100%
```

```
src/message/src/message/__init__.py                                 1      0   100%
src/message_impl/src/message_impl/__init__.py                       5      0   100%
src/message_impl/src/message_impl/_impl.py                         80     24    70%   31-3
-------------------------------------------------------------------------------------------
TOTAL                                                             187     53    72%
```

All Tests (local):

```
src/inbox_client_impl/src/inbox_client_impl/__init__.py            10      3    70%   18-2
src/inbox_client_impl/src/inbox_client_impl/_impl.py               88     23    74%   46-4
src/inbox_client_protocol/src/inbox_client_protocol/__init__.py     3      0   100%
src/message/src/message/__init__.py                                1      0   100%
src/message_impl/src/message_impl/__init__.py                      5      0   100%
src/message_impl/src/message_impl/_impl.py                        80     14    82%   31-3
-------------------------------------------------------------------------------------------
TOTAL                                                             187     40    79%
```

I believe the particular settings for the CircleCI run are also relevant:

```
unit_test:
    docker:
      - image: cimg/python:3.11
    steps:
      - attach_workspace:
          at: . # Attach the workspace persisted from 'build'
      - run:
          name: "Activate Venv and Create Test Results Directory"
          command: |
            source .venv/bin/activate
            mkdir -p test-results/unit # Create directory for JUnit XML
      - run:
          name: "Execute Unit Test Suite (pytest + coverage)"
          command: |
            source .venv/bin/activate
            # Run pytest, collecting coverage, excluding integration tests
            # Ensure mocks are used for external calls in these tests
            pytest . --cov=src --cov-report=xml --cov-report=term \
                     -m "not integration" \
                     --junitxml=test-results/unit/junit.xml
      - run:
          name: "Run Static Analysis (mypy)"
          command: |
            source .venv/bin/activate
            # Target specific source and test directories for mypy
            uv pip install types-requests
            mypy src tests --explicit-package-bases
      - run:
          name: "Enforce Coverage Threshold and Generate Reports"
```

```
command: |
  source .venv/bin/activate
  # Use coverage CLI now that pytest has run it
  coverage report --fail-under=70
  coverage json -o test-results/unit/coverage.json # Generate JSON report
  coverage html -d test-results/unit/htmlcov
```

... and we can clearly see that there are discrepancies between what gets run in CircleCI and what we see locally.

Overview of Component Changes

Upon running `git diff --stat -p 4277bfd2c872eb85f84c68ff953cbd7311c4671f e8e4dcb7ab5c70031f572c8bc18707980c0caa8d`, our output is quite large (~1400) lines and it is omitted for convenience.

Nevertheless, the most significant area of exploration is our inbox_client_impl. Our message_impl also has a few interesting areas we could check out; however, coverage is the same across PRs in CircleCI due to very minor style/whitespace changes, and no logic changes.

As previously stated:

> There is a further discrepancy caused by how we handle authentication in interactive (auth0 logins) vs environment (environment variables injected via CircleCI context) settings. This leads to tests running differently.

Old:

- Only supported environment variable-based auth (for CI) and file-based auth (for local).
- No explicit support for forcing interactive login.

New:

- GmailClient now takes an interactive flag.
- run_interactive_flow and save_token helper methods added.
- Auth flow order: interactive (if forced) → env vars → token file → fallback to interactive.
- More robust error handling and logging.

Coverage Impact:

- Decreases coverage because we do not test the interactive flow (or explicitly omit it from tests) in CircleCI. Causes a big discrepancy felt along the project.
- However, all authentication branches are now testable (and mockable) via the interactive flag.

Overview of Testing

The test suite for inbox_client_impl (test_inbox_client.py) underwent significant improvements in this PR.

Old:

- Tests were more monolithic, with some coverage for environment variable authentication and token refresh.
- The test for authentication via environment variables was present, but there was limited granularity in testing individual methods (send, delete, mark-as-read, etc.) for both success and failure.

New:
Granular Test Functions:

- Each method of GmailClient (send, delete, mark-as-read) now has its own dedicated test for both success and failure cases. This includes:
  - test_send_message_success and test_send_message_failure
  - test_delete_message_success and test_delete_message_failure
  - test_mark_as_read_success and test_mark_as_read_failure

Improved Mocking:
The use of MagicMock and patching is more systematic, ensuring that external dependencies (like the Google API) are fully mocked.

Error Handling Coverage:
Tests now explicitly check that exceptions in API calls (e.g., send, delete, modify) are handled gracefully and return the expected boolean result.

Authentication Branches:
The test for environment variable authentication (test_init_with_env_vars) remains. This keeps CircleCI covered, *but definitely causes problems between environments*.

Removed/Refactored Tests:
Some older, less focused tests (e.g., for token refresh from file) have been removed (Coverage reduction).

Coverage Impact:
Overall we see increases in coverage thanks to these additions.

- By testing both success and failure for each method, more branches in the implementation are exercised.
- The explicit testing of exception handling ensures that error-handling code is not left uncovered.
- The environment variable authentication path is well-covered. However, again, the interactive authentication flow is still not tested in CI.

Concluding Thoughts

Overall, the new structure makes it much easier to add future tests for new features or edge cases, and to mock out dependencies for true unit testing. There are several improvements that could be done. In particular:

1. Adding Tests for the Interactive Authentication Path: need I say more.
2. Test Token File and Refresh Logic: Some of the removed/refactored tests for token refresh from file could be reintroduced in a more focused way, using mocks to simulate expired tokens and file presence/absence.

Above everything, I would like to hear your opinion on my claims made in the Coverage Extra Credit > Opinion Section.